

Barry Norton and Nenad Stojanovic (Editors)

Proceedings

of the

6th International Workshop on Semantic Business Process Management (SBPM2011)

In conjunction with the

8th Extended Semantic Web Conference (ESWC2011)

May 29, 2011, Heraklion, Greece

Organizing Committee:

Barry Norton (Karlsruhe Institute of Technology)

Nenad Stojanovic (FZI Research Center for Information Technologies Karlsruhe)

Workshop URI: <http://sbpm2011.fzi.de>

Table of Contents

Program Committee	iii
Introduction	iv
Semantic Technologies for the Project Management Life Cycle Improvement	1
<i>Birgit Dippelreiter, Michael Pöttler</i>	
Information Resource Recommendation in Knowledge Processes	9
<i>Tadej Štajner, Dunja Mladenić, Marko Grobelnik</i>	
Querying Semantically Enriched Sensor Observations.....	17
<i>Jean-Paul Calbimonte, Hoyoung Jeung, Oscar Corcho</i>	
Service Adaptation Recommender in the Event Marketplace: Conceptual View	18
<i>Yiannis Verginadis, Ioannis Patiniotakis, Nikos Papageorgiou, Roland Stuehmer</i>	
Collaboration in Open Source Software Development: The ALERT Approach	26
<i>Ljiljana Stojanovic, Sinan Sen, Jun Ma</i>	

Introduction

The 6th International workshop on Semantic Business Process Management continues the tradition of previous workshops from this serial: collecting and discussing new semantic-based approaches and ideas for a more efficient BPM. This year the most important issue is the event-driven nature of business processes that introduces extreme agility in business, especially in the case of informal business processes. In other words, business processes are ready to sense and respond on different external and internal events and change the flow on the fly. This requires developing methods suitable for gathering events from the environments, including unique format for their description, efficient combination of these events and the presentation/visualization of results. Indeed, the topic is very promising, as this Gartner prediction says:

By 2013, dynamic BPM will be an imperative for companies seeking process efficiencies in increasingly chaotic environments.

IT organisations are striving to become better aligned with the demands placed on the business. Pressure to reduce the latency of change in business processes is driving a need for more dynamic and systematised measures. Adopting a more dynamic form of BPM, which focuses on enabling process changes to occur when and as needed will enable organisations to better respond to unanticipated change requirements in business processes, and to handle process changes more effectively.

More than a half of papers are dealing with the different approaches for managing this dynamics in business processed by using semantic technologies. Rest of the papers is targeting the dynamicity from the modeling point of view (design time). Therefore, this workshop is a nice opportunity for exchanging ideas and starting new cooperation in this domain, as it was in the past.

Program Committee

Agata Filipowska

Ana Karla Alves de Medeiros

Andreas Abecker

Carlos Pedrinaci

Darko Anicic

Florian Lautenbacher

Jan Mendling

John Domingue

Jorge Cardoso

Juhnyoung Lee

Ljiljana Stojanovic

Liliana Cabral

Markus Nüttgens

Michel Klein

Oliver Thomas

Oscar Corcho

Roxana Belecheanu

Tommaso Di Noia

Witold Abramowicz

York Sure

Yuxiao Zha

Semantic Technologies for the Project Management Life Cycle Improvement

Birgit Dippelreiter, Michael Pöttler

Vienna University of Technology, Institute of Software Technology and interactive
Systems, Favoritenstrasse 9-11,
1040 Vienna, Austria
{birgit.dippelreiter, michael.poettler}@ec.tuwien.ac.at

Abstract. Existing knowledge of finished projects, such as documents, lessons learned or competences of project members, has to be reused to minimize project related shortcomings and to improve the quality of current project management. Our studies, conducted with 11 IT companies, reviewed the status of current project management practices and confirmed such failings. Thus, these shortcomings are represented as scenarios, which build the base for the project management ontology proposed in this paper. By incorporating semantic technologies to increase the quality of project management processes, our ontology includes concepts about projects, tasks, documents, persons and their competences as well as lessons learned. Our approach will be implemented and evaluated by means of a software prototype, where we extend the open source project management system ‘dotproject’ with the described ontology, semantic annotations and a semantic search function.

Keywords: ontology, semantic project management, ontological engineering, knowledge base

1 Introduction

Despite the fact that project management (PM) is a practice realized in almost every company to achieve predefined project goals and constraints, PM systems still contain lots of shortcomings. Cerpa and Verner [3] identified the most common failure factors by analyzing 70 failed projects, such as management and time constraints or risks which were not adequately controlled. Emam and Koru [5] and the Standish Group [14] refer to a statistical overview of failures and identify a different set of factors, such as lacks of user input or executive support.

Another identified deficit is that most companies do not reuse existing knowledge of finished projects. In fact they do not archive information in a central and well-structured storage [4]. In addition, current project management systems mainly support the ongoing phase and do not consider the initiating and closing phase of a project management life cycle. Honda et al. [11] define the most problems within IT projects during the initiating phase. Thus, project management systems are not

flexible enough to react, e.g., on lessons learned, which would increase ongoing projects within the PM life cycle.

The PM life cycle consists of an initiating, an ongoing (planning, executing and monitoring & controlling) and a closing phase [12] (depicted in figure 1). While the initiating phase contains the definition of a new project or the composition of the project team, the closing phase includes the finalization of all project relevant activities and the archiving of all relevant project information. By contrast, the ongoing phase deals with, e.g., work packages including requirements, project execution and controlling costs.

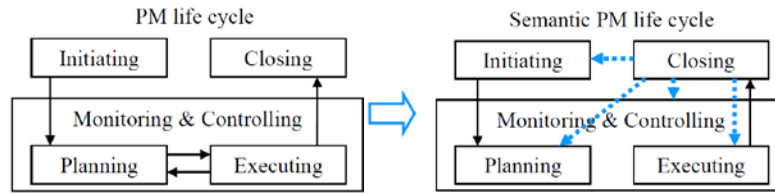


Fig. 1. PM's life cycle [12]

Aim of our project SemProM – Semantic based Project Management - is to evaluate whether “semantic technologies could improve current project management processes”. Thus, an open source project management system will be enhanced with semantic technologies; more precisely a project management ontology, semantic annotations and a semantic search function. In addition, the stored information is semantically linked and archived in a central storage. The outcome is to obtain a semantic knowledge base to query project relevant information improving the project management life cycle (shown in figure 1). Thus, knowledge of closed projects can be used for the initiating or planning and executing phase of other projects.

Our first assumption is that due to the use of semantic technologies knowledge of projects is easier detectable and can be retrieved for further projects. In addition, our second assumption states that the system is able to react on unforeseen circumstances such as ‘is there any other employee in my company with the same competencies like the ill project member to ensure a positive project progress?’. As mentioned above, due to the fact, available information is linked within the knowledge base, e.g., documents, employees and lessons learned, more precise search results can be achieved. Furthermore, by using the standardized exchange format RDF (Resource Description Framework), the interchange of relevant information gets simplified; for instance, to enable the connection to another system.

In this paper we introduce on the one hand the engineering approach of the project management ontology and on the other hand a description of the PM ontology including its concepts and relations.

This paper is structured as follows: Section 2 displays a short overview on the work steps of the project SemProM and the ‘Related Work’. The methodological approach concerning the ontology engineering method is depicted in section 3. Section 4 illustrates the ontology, its most important concepts and relations as well as the term definitions. The ‘Proof of Concept’ of the ontology is described in section 5. Conclusion and ‘Future Work’ are resumed in the last section.

2 Background

The work steps within the project SemProM are handled as follows: First, to evaluate our research question, we conducted eleven face-to-face interviews with IT companies to identify the strengths and weaknesses of current PM solutions with the main results [4]: Companies miss a central, well-structured and accessible storage; Project managers and members miss an up-to-date cockpit of ongoing projects, including all relevant project items, e.g., milestones and costs; and, most of the companies do not reuse knowledge of finished projects to improve ongoing and further ones.

We decided to follow a qualitative approach by using a questionnaire with 27 questions. A detailed description of the interviews and their results is given in the paper Dippelreiter et al. [4].

Second, based on these outcomes, we specified scenarios, which cover these results. For instance, one scenario named “team suggestions” deals with recovering of finished project knowledge of team work and collaboration of project members. Third, to tackle the identified outcomes, we implemented a PM ontology containing information regarding projects, documents (information), persons, competences and lessons learned. Fourth, a software prototype enhanced with semantic technologies will be implemented. And fifth, the evaluation of our research question will be accomplished by means of the prototype.

While developing the PM ontology a literature review regarding already existing ontologies was done. For our purpose the main focus lies on project management and date and time ontologies. Thus, e.g., the PROMONT ontology [1] formalizes the typical elements for project structuring. The further development of this ontology, divided into a project and product ontology, was done as PR²ONTO in the project PERMETER [8]. Due to the fact that PROMONT covers different project management specifications, especially the DIN 69901 [1] it is too general for our purpose, identified by the scenarios. Regardless, we looked closely to the PROMONT ontology and used, e.g., the properties time and responsible person.

In addition, the Semantic MediaWiki [9] has to be mentioned as well. This Wiki contains a project management ontology and provides the collaborative knowledge management. Contrary to PROMONT it fits to certain requirements concerning the scenarios. For instance, properties regarding planned and actual start dates or the concept structure of Resource and Person (Person is a subclass of Resource).

Other ontologies, which influenced our building approach, are included in the Semantic Desktop Gnowsisi [13], especially date and time. During the implementation process of the PM ontology we realized that a specific date and time ontology is not needed. For our purpose just date and time properties related to tasks are sufficient.

Due to the fact we categorized the competence part of our ontology only in a first structure, we identified the categorization of [10]. For further development competence ontologies, like the ontology of Biesalski [2], can be used.

3 Methodological Approach

A literature study regarding the ontological engineering approach shows Gomez-Perez et al. [6] giving an overview of methods for building ontologies. The most important ones, which influenced our ontological engineering approach, are Uschold and King [15] and Grüninger and Fox [7].

For the implementation of our ontology, we used a mix of both approaches. While the engineering approach of Uschold and King [15] is more related to modeling and building up an ontology within a special domain, Grüninger and Fox [7] use a more formal and technical approach which is query centered. By merging both approaches, we obtain a detailed documentation of the ontology, its concepts, properties and relations, its term definitions and an illustration how to extend the ontology. In addition, scenarios are used as ontology requirements as well as for the evaluation of the prototype. Beyond that informal and formal competency questions specify the ontology in more detail and simplify the implementation of the prototype.



Fig. 2. SemProM's ontological engineering approach

Depicted in figure 2, our engineering approach contains the following steps in detail:

1. **Identify purpose:** The ontology will cover current PM problems; e.g., relations between tasks, their needed competences, project members and lessons learned
2. **Motivating scenarios:** We identified three scenarios, which deal with current weaknesses of project management solutions. They describe both, the requirements of the ontology as well as possible solution processes.
3. **Informal competency questions:** These questions, derived from the scenarios, are natural language questions. They will be answered by querying the formal questions testing the prototype.
4. **Building the ontology & terminology:** Based on the scenario requirements and the informal competency questions we identified the ontology concepts, properties and relations and their term definitions.
5. **Formal competency questions & Evaluation:** To write the competency questions in a formal way, we used SPARQL, an RDF query language. In addition, to evaluate our ontology we simulate the scenarios by using the prototype and querying the questions.

To evaluate our ontology we reviewed the consistency by using the consistency check of our used ontology editor TopBraid¹, the Website consVISor² (a tool for checking the consistency of OWL ontologies) and

¹ TopBraid Composer - http://www.topquadrant.com/products/TB_Composer.html

² consVISor - <http://173.14.188.57:8080/consvisor/>

the W3C Validation Service³. The main evaluation will be accomplished by testing the prototype with the predefined scenarios.

6. **Documentation:** To keep track of the ontology, we documented the concepts, properties and relations and their term definitions. In addition, the modularity of the ontology and the interfaces are described. Thus, the ontology is easily extendable and can be used for any other project management system.

4 Ontology & Terminology

This section describes the main structure of the developed PM ontology including the most important examples of concepts and relations, depicted in figure 3.

As already mentioned, during the implementation process of the ontology we attempt to cover shortcomings of current PM solutions within the field of IT. As a result, the PM ontology deals with the necessary domains of projects, tasks, persons, competences, information and lessons learned and their relations. For instance, the relation between persons, tasks, function and competences plays an important role. Likewise lessons learned concerning any topic related to a task. Due to the requirements are resulting out of the scenarios, it was possible to restrict the domains covered by the ontology. But if new scenarios have to be handled by the PM ontology, it can easily be extended and adapted.

To start the explanation of figure 3, first the terms of the individual concepts are defined; followed by an explanation of the relations between them.

Task: A task can contain several sub-tasks. The task, which is no sub-task of another task can be identified as project; while sub-tasks can be termed as work packages, deliverables or tasks. Each task is related to a person, who is responsible for the task. In addition, the task is related to at least one function, which describes the activities within that task. To each function at least one person is related. Thus, the relation between task, function and person is handled.

Person: A person is any employee. A person can be assigned to a task and within the task to an activity. In addition, a person works well with another person and has several competences.

Information: The concept **Information** contains any kind of project relevant documents; e.g., project plan or documentation of a software module.

Competences: The concept **Competences** contains any kind of competences of a person or a task. The classification is as follows: methodological competence, professional competence, personal competence and social competence.

MethodologicalCompetence is the qualification to use specific learn and working methods. While **PersonalCompetence** is, for instance, accuracy or to work under pressure. In the PM ontology the concept **ProfessionalCompetence** contains competences in the field of IT. But if necessary, any other domain can be added. The last sub-concept **SocialCompetence** includes social, emotional and intellectual skills.

³ W3C Validation Service - <http://www.w3.org/RDF/Validator/>

LessonsLearned: LessonsLearned is any kind of information of experiences or hints aggregated during an existing project, e.g. a bad team composition or problems during a development process, which helps to improve further projects.

Function_Relation: This concept is sort of a utility class. It enables the relation between the concepts Task, Person and Function. A task implies some functions/activities, which will be assigned to at least one person.

Function: The concept Function implies the activity within a task. Each task requires one or even more fields of duties.

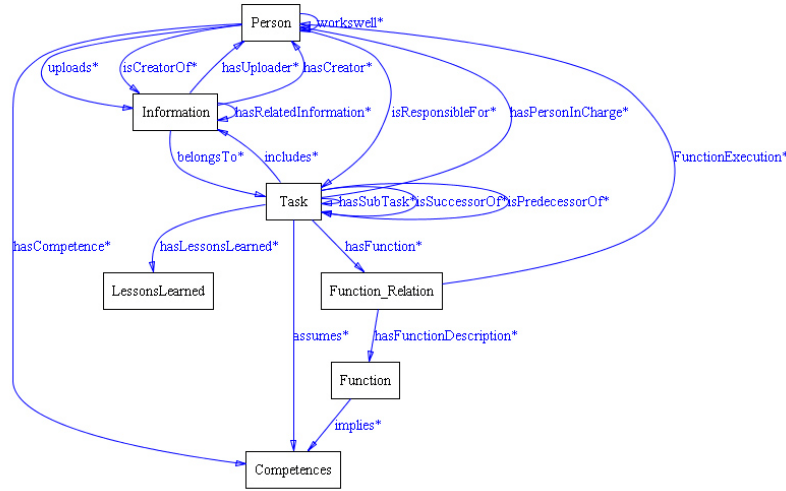


Fig. 3. Main concepts and their relations of the project management ontology

According to the term definitions, the most relevant relations between the concepts can be explained as follows:

A task is able to contain sub-tasks (**hasSubTask**). This implies a hierarchical structuring of a project/task into smaller working areas. If a task depends to another task it can be a successor (**isSuccessorOf**) or a predecessor (**isPredecessorOf**) of a task. In addition, a task has a person, who is responsible for it (**hasPersonInCharge**, inverse of **isResponsibleFor**). Furthermore, a task includes several documents (**Information**; **includes**, inverse of **belongsTo**) and implies lessons learned (**hasLessonsLearned**). To implement each task, the task assumes competences, which have to be fulfilled by persons. In addition, a task has a function, which will be accomplished by a person (**Function_Relation**). Moreover, the function requires competences.

Additionally, a person is able to create (**isCreatorOf**, inverse of **hasCreator**) and upload (**uploads**, inverse of **hasUploader**) documents. These may also be related to other ones (**hasRelatedInformation**).

By itself the ontology shall not be considered as finished. Although, it covers all necessary aspects of our scenarios. It can be rated as foundation for further work. Thus, the individual parts, for instance, competences and lessons learned, can be

easily extended. For instance, a project management system within the domain of architecture is needed, the ontology can easily be enhanced with the structuring of professional knowledge of, e.g., architecture.

5 Proof of Concept

In order to evaluate the PM ontology and in further consequence to answer the question whether “semantic technologies could improve current project management processes” a prototype consisting of the open source project management system dotproject (<http://www.dotproject.net>), a mysql database and the PM ontology is implemented.

To evaluate our research question, the scenarios had to be implemented as well. Therefore, the preliminary work was the input of instances as well as the definition of the criteria catalog deduced from the scenarios. The foundation instantiation of the PM knowledge base was done manually by using the user interface of the prototype. Next step will be the testing of our prototype by going through the criteria catalog. Our test persons will be project managers working within the domain of IT. During the test runs, test protocols including topics such as handling, scenario processes and notions will be created. By means of the test results, the research questions will be evaluated.

Primary objective of this proof of concept is geared towards the evaluation of our research question. In further work an automation of data input into the knowledge base is planned.

6 Conclusion and Future Work

In this paper we outlined the project management ontology elaborated within the project SemProM, which contains concepts, relations and properties of projects, tasks, persons, competences, documents and lessons learned. Scenarios, which reflect the requirements of the ontology, are the basis of the PM ontology. These scenarios cover shortcomings, conducted by interviews with project managers of 11 IT companies concerning their project management solutions.

In addition, an in-depth study of existing project management and time/date ontologies was carried out. Based on our required needs we decided to implement an individual PM ontology due to the fact that the evaluated ones are partially too general for our purpose. Relating the ontology engineering approach we discussed Uschold and King [15] and Grüninger and Fox [7]. As a result, we combined both approaches. While giving an overview of the main structure of the PM ontology, term definitions of the concepts and the relations between the concepts were explained.

The next steps of this project are the implementation of the software prototype and the evaluation applied on the predefined scenarios. The assumptions are: Providing an effective storage and linkage of project relevant knowledge within a central semantic knowledge base as well as an efficient semantic search through the stored knowledge to improve the initiating and closing phase of the project management life cycle. In

addition, a kind of cockpit that provides an overview of all projects relevant information is enabled.

Acknowledgement

The project SemProM is funded by a three year fellowship from the Austrian funding agency FFG (the Austrian Research Promotion Agency).

References

1. Abels, S., Ahlemann, F., Hahn, A., Hausmann, K., Strickmann, J.: PROMONT – A Project Management Ontology as a Reference for Virtual Project Organizations. Lecture Notes in Computer Science, Springer Verlag, p. 813 – 823 (2006)
2. Biesalski, E.: Unterstützung der Personalentwicklung mit ontologiebasiertem Kompetenzmanagement. Dissertation. Universität Karlsruhe. (2006)
3. Cerpa, N., Verner, J.M.: Why did your project fail? Communications of the ACM. Volume 52, Issue 12, p. 130 – 134 (2009)
4. Dippelreiter, B., Grün, C., Pöttler, M.: A ‚State of the Art‘ Evaluation of PM-Systems Exploring their missing Functionalities. In: Proceedings of the 5th International Conference on Project Management. (2010)
5. Emam, K.E., Koru, A.G.: A replicated Survey of IT Software Project Failures. IEEE Software. Volume 25, p. 84 – 90 (2008)
6. Gomez-Perez, A., Fernandes-Lopez, M., Corcho, O.: Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. Springer. (2004)
7. Grüninger, M., Fox, M.: Methodology for the design and Evaluation of Ontologies. In: IJCAI’95, Workshop on Basic Ontological Issues in Knowledge Sharing. (1995)
8. Hahn, A., Hausmann, K., Häusler, S., Strickmann, J.: Using Ontologies to Model and Understand Product Development. Artikel in IBIS – Interoperability in Business Information Systems, Heft 5. (2007)
9. Hansch, D.: Practical applications of Semantic MediaWiki in commercial environments – Case Study: semantic-based project management. In: ESTC 3rd Annual European Semantic Technology Conference (2009)
10. Heyse, V., Erpenbeck, J.: Kompetenztraining, 64 Informations- und Trainingsprogramme: Schäffer-Poeschel Verlag Stuttgart, 2004
11. Honda, Y., Ogawa, J., Tatebe, K.: Committee meeting for exit criteria at early phase of IT project. In: Proceedings of the Society of Project Management ProMAC Symposium, p. 132 - 137 (2009)
12. Project Management Institute (2008): A Guide to the Project Management Body of Knowledge (PMBOK). 4th Edition. Newtown Square
13. Sauermann, L., Grimnes, G., Kiesel, M., Fluit, C., Maus, H., Heim, D., Nadeem, D., Adrian, B., Dengel, A.: Semantic Desktop 2.0: The Gnowsiss Experience. In: Proceedings of the ISWC Conference, Springer (2006)
14. Standish Group.: Chaos Report (1994)
15. Uschold, M., King, M.: Towards a Methodology for Building Ontologies. In: workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95. (1995)

Information Resource Recommendation in Knowledge Processes

Tadej Štajner, Dunja Mladenić, Marko Grobelnik

Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
{tadej.stajner, dunja.mladenic, marko.grobelnik}@ijs.si

Abstract. This paper proposes a framework and an implementation for proactive information resource delivery, based on knowledge processes. We focus on providing a light-weight implementation and deployment scenario resulting in a desktop application that presents a ranked list of information resources, such as documents, web sites or e-mail messages, that are considered to be most relevant in that point in time. Evaluation on several event, action and process models suggests that defining actions based on clustering of events yields the best recommendations.

Keywords: Process modeling, Information delivery, Clustering, Action mining, Process mining

1 Introduction

Given the scenario of knowledge workers in an enterprise whose workflows consist of accessing various resources, such as web pages, documents and e-mail messages, we propose a prototype for process-based prediction and its implementation. The premise of this work is that we can learn a model from patterns of events within the usage these patterns, which we refer to as the knowledge process model, and use the learned model in a way that would streamline the workflow by suggesting information resources to the user before he needs to retrieve them [10]. For the purpose of this paper, we use the term *knowledge process model*, which corresponds to a set of patterns that describe how a knowledge worker is using various information resources [1]. We construct the knowledge process model by learning from existing event logs under the assumption that the event logs are generated by an underlying knowledge process model.

We capture these events in an on-line fashion through workspace instrumentation and logging infrastructure. For example, the events consist of accesses of search engines and web sites, documents on the desktop and usage of e-mail. We use the resulting event stream to both learn knowledge process models from usage patterns, as well as react with suggestions for information resources given live data, which we distill to contain three basic components: text, social network, and time. The implementation suggests information resources that are considered to be the most

relevant in that point in time, according to the knowledge process model learned from users' previous behaviour.

We decompose the problem into subproblems that form a *knowledge process framework* that enables us to consider process mining on top of primitive events. We outline various *event*, *action* and *process* models that fit various scenarios and patterns that we have encountered in our experiments. The *event models*' role is to fit sequences of primitive events with various flavors of content, semantics and social network information and encode them into feature vectors. The *action models* take these sequences of feature vectors and transform them into sequences of actions, which can then be processed by *process models* to act as predictors of most likely future actions, from which we can then apply a ranking on a candidate set of information resources which have the same type of representation as individual events.

2 Related Work

Process mining is often used to uncover dependencies, control flows and patterns in a given business process execution log, encoding the model in eEPC diagrams or Petri nets [2]. Whereas these sort of approaches were built for the purpose of discovering process knowledge that enabled organization optimization, we are targeting the use case of pro-active information delivery. As presented in Holz et al. [3], such approaches mainly vary in two dimensions: process support and information delivery, meaning that some information delivery approaches work on top processes ranging from ad-hoc to strictly-structured, and having from light- to heavy-weight information delivery mechanisms. In our domain, having light-weight modeling is important to simplify deployment and adaptation. Our design goal is to maximize information delivery performance while learning from usage logs alone. Besides enterprise knowledge processes, systems that learn while monitoring were also proven successful in the e-learning domain [4] with work-integrated learning [5].

3 Knowledge process model framework

We have designed and implemented a framework for prediction on top of knowledge processes that solve our issues that we have encountered with designing prediction use cases with classic process mining models. In our experience, process mining expects well-defined atomic actions as its input - which is a requirement which may not be easily satisfied in some domains, such as most knowledge work [6]. The data that we are dealing with is a natural example of the TNT (text, network, time) framework [8]: the data points are events carrying temporal information, content and a social network component. To fit into that model, use a framework that is designed to handle that situation with to support process mining with semi-structured data [7].

We decompose the information delivery problem in knowledge processes into a framework with three separate subproblems, each solveable with multiple approaches. To better illustrate the design decision behind the three distinct steps, let us start from the final step: the *process model*. In business process modeling, a process model is composed describes patterns of atomic actions and can be instantiated as a sequence of actions. A process mining model maps from an action history to a probability distribution across possible following actions. We also need to transform the TNT events to actions - an *action model*. However, directly constructing an action model to map from TNT events to process actions is not always possible nor practical, since the content within the events may not have the same properties across different domains. For this purpose we define the *event model*, which represents the feature construction phase from the raw TNT events.

The prediction scenario uses all of these three steps in the following setting: given a user's history of TNT events and possible resources that will be used, transform both the history and the future candidates into actions. Next, use the process model to evaluate each candidate's probability of appearing given the observed history. This information is then used in evaluation as a ranking score for information resources.

3.1 Input data

TNT events may contain the information on the actor, textual content, social network information, event metadata and the time of execution. We partition the log of events into sessions, which represent instances of knowledge process executions.

3.2 Feature construction with event models

A **vector space event model** represents the TNT event as a vector of features, derived from the event's text, social network and metadata. This representation is richer and allows the use of machine learning techniques, appropriate for vector-space data. For text-derived features, we apply a TF-IDF weighing scheme.

This model can be extended by also encoding the information from neighbouring events. The sequential addition to the simple vector space event model enables event representations that captures some session information, resulting in a **session vector space** model, which takes into account the positioning, but only the fact that a step has been executed in the same session within a scope, regardless of order.

3.3 Action models

From this point on, we only consider documents as feature vectors and pay no attention to the nature of the features themselves. However, as basic process models still map from a space of action-to-action transitions to conditional probabilities, we still need to map sequences of documents into sequences of actions.

Independent feature actions. There are several ways to avoid high dimensionality of actions: one is to treat event features independent and consider them as individual actions themselves. However, this leads us to an issue where we have multiple possible actions per event, which we need to somehow interpret. We solve this in the following fashion: given a pair of consecutive event vectors, consider at each possible feature pairing from these two vectors as a potential sequence. For instance, given a sequence of two consecutive events $d_1 = \{a = 0.7, b = 0.7\}$ and $d_2 = \{x = 0.7, y = 0.7\}$, we translate this to action sequences of two consecutive actions: (a,x) , (b,x) , (a,y) and (b,y) . In other words, each of these action sequences is a possible interpretation of (d_1, d_2) in this action model. The consequence of this is that for some concrete dataset, the feature space of events is the same as the feature space of actions. This is one way to model the fact that an event may have many features. The trade-off that this model provides is that we keep the space of possible actions reasonably low-dimensional, but assumes conditional independence of features. In this model, multiple actions may represent a single event. To resolve this, we choose the most likely interpretation.

Clustering-based actions. As has been suggested in [6], one way to define actions is by performing various types of clustering on the event feature vectors. We have also examined the effect of the k parameter in k-means clustering on the predictive power of the model, demonstrating that it is possible to define clusters of events as actions and still retain good perplexity [7]. This model allows us to precisely control the dimensionality of the action space by varying the k parameter. Upon mapping events into actions, we classify the new set of events into actions via a centroid classifier.

3.4 Process models

The core of the framework are the process models. They estimate the probability that a certain action will take place conditioned by the last couple of observed actions. There are several aspects of process models that we can also consider when developing a forecasting solution.

Smoothing of the process model is necessary because sparseness of data is a real problem. The reason for that is in order to successfully compute a probability of a sequence, none of the sub-sequence probabilities must turn out to be zero. However, if there is no evidence for a particular action following a particular sequence, it does not mean that that sequence will also not occur in test data. For that purpose we employ Laplace smoothing.

Even though the framework allows any possible combination of event, action and process models, some combinations yield better results than others. It is also worth explicitly noting that this framework enables two different ways to encode dependencies in the knowledge process: either via features in sequential- and session-based event models, or explicitly within the core process model. We allow both scenarios, since interesting dependencies might exist either on the level of individual features, or between actions, having a higher level of abstraction.

4 Implementation

We have implemented prediction that fits into the desktop environment of knowledge workers. The scenario is the following: imagine a desktop widget that tracks the current state of the user's workspace with most recently accessed resources. The widget itself is rendered as a ranked list of information objects that are considered most relevant for the user. We use the following semantic properties as features for individual events: bag-of-words of document content, social roles of participants (inside vs. outside of organization, manager, developer, researcher, private vs. multiple people, single vs. multiple organizations) and event metadata (type of event, type of media).

Multiple criteria are used to ensure redundancy for cases where the user is not in any context or the user has just started his session and does not have any recently accessed resources. To use the knowledge process infrastructure as a recommender system, we combine the following criteria for ranking:

- The probability of the candidate information resource d_i given the history: $P(d_i | d_{hist\ 0...}, d_{hist\ j})$;
- The similarity of the candidate information resource to the most recently observed information resource: $sim(d_i, d_{hist\ j})$;

The implemented front-end is a simple graphical user interface, displaying an ordered list of most probable information objects given the current state of the user, as shown in Figure. Given the live events coming from the monitoring infrastructure, the information delivery framework converts these events into feature vector documents and then action sequences which represent the history. Then, it selects a set of possible interesting information resources with standard text retrieval techniques and evaluates the probability of each known information object given the observed history.

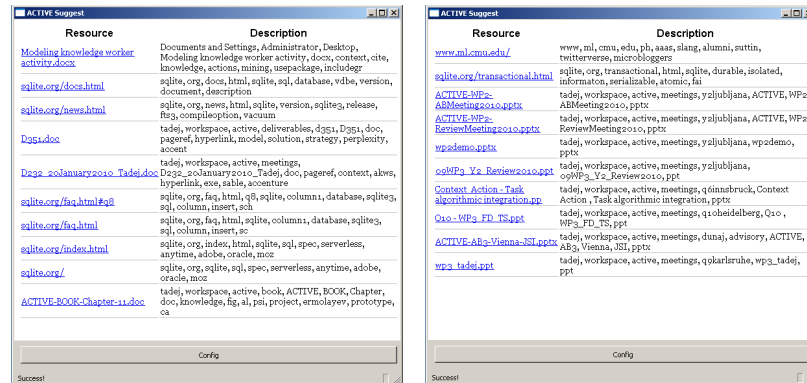


Fig. 1. Examples of recommendations for various situations in the knowledge process

Figure 1 shows some examples of recommendations, offered by the service. The first image shows the state of the recommender after reading a documentation of a back-end system for process mining, which is reflected by similar content types being offered. The second image demonstrates another point in the knowledge process, showing recommendations after opening a technical presentation on machine learning. It offers various related documents, as well as web sites that might contain useful information in that situation.

5 Experiments

In our experiments, we have performed evaluation by constructing combinations of models with one set of data and evaluating them with another subset using ten-fold cross-validation to account for generalization error: we ran the experiment ten times, using nine partitions as training data and the remaining one as a test dataset. Given this setup, we can observe several evaluation metrics, relevant to evaluation of models.

We evaluated various knowledge process model configurations by measuring how high do they rank the actual events in the test set. In order to conduct this experiment, we simulate the prediction behaviour by querying the system with partial sequences to which we already know the correct TNT event that will follow and then check how high does that particular event rank in the prediction output. In other words, we are measuring how well can this system predict the actual workflow. We then report the average reciprocal rank for a particular combination of models and average number of predictions that score in the top 20, which are the key indicators in our experiment.

Since we don't expect the process model alone to produce a good ranking by itself, we run it in combination with a baseline: we have chosen the similarity between the most recent document and the candidate document for that purpose. We merge the two rankings by multiplying the similarity with a probability coefficient of the candidate. This is done in order to see whether having a process model can be approximated by just biasing the ranking towards resources, more similar to the last accessed resource. The intuition behind this using the context model is when a knowledge worker is working in a context, we assume that the resources, belonging to a context, tend to be similar to one another.

We have evaluated on the knowledge worker logs. Since the TNT events have no pre-defined actions, we use the vector space event models – *IDF* and *SessionIDF*. They all use the TF-IDF weighing scheme for features, but differ in the way that the features are generated: whereas *IDF* contains only the features from a particular event, *SessionIDF* also encodes between the features of the events in the same session. Since we are using feature vectors representations which tend to have multiple features, this means that in order to generate a manageable number of distinct actions, the action model can not be an identity, but one either *Independent* or *Clustered*. For process models, we experimented on either using an identity transform

which assumes uniform processes as a baseline or a bi-gram Markov model model with Laplace smoothing.

Table 1. Experiment results with various combinations of event, action and process models for the desktop use case.

Event Model	Action Model	Process Model	Reciprocal rank	Percentage in top 20
IDF	Independent	Independent	0.0612	0.2220
IDF	Independent	Laplace	0.0803	0.2377
IDF	Clustered:10	Independent	0.0794	0.2697
IDF	Clustered:10	Laplace	0.1076	0.3485
IDF	Clustered:30	Independent	0.0853	0.3081
IDF	Clustered:30	Laplace	0.0797	0.2490
SessionIDF	Independent	Independent	0.0774	0.2895
SessionIDF	Independent	Laplace	0.0750	0.2674
SessionIDF	Clustered:10	Independent	0.0756	0.2807
SessionIDF	Clustered:10	Laplace	0.0701	0.2384
SessionIDF	Clustered:30	Independent	0.0832	0.3013
SessionIDF	Clustered:30	Laplace	0.0874	0.3051

We have ran the experiments on the dataset, gathered in two separate two-month time periods, totaling 31882 events. Here, the events consist of web browsing events, editing of Microsoft Office documents and working with e-mail. Results, shown in Table 1, show the following: best performing configuration is the one with plain TF-IDF features of events, defining a clustered action model with few distinct clusters as actions and a process model with Laplace smoothing. The difference here is that in this case, we were able to better capture sequential dependencies on the action model level by the process model, rather than encoding the dependencies into the features themselves, suggesting that the underlying process is relatively simple. This could be explained by the fact that the Accenture dataset was generated by several tens of thousands of different users in various roles, whereas the BT dataset was generated by three users. All in all, we are able to place the correct information resource in the top-20 list roughly on over one third of occasions.

6 Conclusion

An important take-home lesson is that many of the issues in process mining lie in transforming the input data into well-defined actions. By developing a framework that integrates the data transformation steps within the mining process itself, we can simultaneously solve the issue of transforming events into atomic actions, as well as learning process models, reducing the preprocessing requirements for implementing a predictive application. We have discovered that encoding the dependency patterns in the feature vectors rather is a valid approach, showing significant improvement from the baseline. However, our use case shows that encoding the dependencies on the level of actions and not features works even better. Choosing whether encoding the dependencies via features or via process models depends on the granularity of the

underlying knowledge process model. When we wish to model on a lower level of abstraction, using feature dependencies can be sufficient. When the patterns can not be easily expressed on the level of features, modeling dependencies via process models can also yield acceptable performance.

Implementation-wise, we have succeeded in producing a simple user interface that does not require that the user interacts with it in any special way to train it, lowering the barrier to usage. For our future work, the clustering-based action definition will include using a complex graph representation of data so that we can avoid flattening the semantic network structure into event features. To take advantage of the structural information and to correctly handle differences in distributions across people, events and resources, present in the multi-relational representation, we will need to employ multi-relational clustering algorithms which are able to handle such datasets. We will also experiment in applying this model in other specific knowledge worker domains, focusing on supporting multiple knowledge workers with personalized models.

References

1. Warren, P., Kings, N., Thurlow, I., Davies, J., Bürger, T., Simperl, E., Ruiz, C., Gomez-Perez, J., Ermolayev, V., Ghani, R., Tilly, M., Bösser, T., Imtiaz, A.: Improving knowledge worker productivity the ACTIVE approach. *BT Technology Journal* 26(2) (2009)
2. van Dongen, B.F. and de Medeiros, A.K.A. and Verbeek, H.M.W. and Weijters, A. and van der Aalst, W.M.P.: The ProM framework: A new era in process mining tool support, *Application and Theory of Petri Nets*, vol. 3536, pp. 444-454, Springer (2005)
3. Holz, H. and Maus, H. and Bernardi, A. and Rostanin, O.: From lightweight, proactive information delivery to business process-oriented knowledge management. *Journal of Universal Knowledge Management*, vol. 2, 101--127 (2005)
4. Lokaiczuk, R., Faatz, A., Beckhaus, A., Goertz, M.: Enhancing just-in-time e-learning through machine learning on desktop context sensors. In: *Proceedings of the 6th international and interdisciplinary conference on Modeling and using context*. pp. 330–341. Springer-Verlag (2007)
5. Rath, A. and Devaurs, D. and Lindstaedt, S.: Studying the factors influencing automatic user task detection on the computer desktop. *Sustaining TEL: From Innovation to Learning and Practice*, 292—307, Springer (2010)
6. Štajner, T., Mladenić, D.: *Modeling Knowledge Worker Activity: Workshop on Applications of Pattern Analysis*, Cumberland Lodge (2010)
7. Štajner, T., Mladenić, D., Grobelnik, M.: *Exploring Contexts and Actions in Knowledge Processes: Proceedings of the 2nd Workshop on Context, Information and Ontologies* (2010)
8. Grobelnik, M., Mladenić, D., Ferlež, J.: Probabilistic Temporal Process Model for Knowledge Processes: Handling a Stream of Linked Text. *Proceedings of SiKDD 2009 (Conference on Data Mining and Data Warehouses)*, 2009.
9. Gomez-Perez, J., Grobelnik, M., Ruiz, C., Tilly, M., Warren, P.: Using task context to achieve effective information delivery. In: *Proceedings of the 1st Workshop on Context, Information and Ontologies*. pp. 1–6. ACM (2009)

Querying Semantically Enriched Sensor Observations: Short Paper

Jean-Paul Calbimonte¹, Hoyoung Jeung², and Oscar Corcho¹

¹Ontology Engineering Group, Departamento de Inteligencia Artificial,
Facultad de Informática, Universidad Politécnica de Madrid, Spain
`jp.calbimonte@upm.es, ocorcho@fi.upm.es`

²School of Computer and Communication Sciences,
Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
`hoyoung.jeung@epfl.ch`

Abstract. The increase that sensor network deployments are experiencing makes them one of the fastest growing sources of data nowadays. They allow live and in-situ data capture and provide high volumes of information about the real world that surrounds us. However their heterogeneity and error-proneness make them hard to use, as we normally require lots of details of each one before their measurements can be combined. We present an ontology-based approach for querying live sensor data observations, enriched with semantic metadata. The metadata is used in two scenarios: (i) for adding contextual (provenance) information to the sensor observations, such as which sensor recorded what, where, when, and in which conditions; and (ii) to help users filter sensor data in their queries. We provide automatic query translation and join of sensor data and metadata, using mappings of the underlying sensor schemas to semantically annotate sensor observations as a new type of metadata. With this approach we can express queries in terms of a high level sensor network ontology, using an extended version of the SPARQL query language that considers window constructs.

Service Adaptation Recommender in the Event Marketplace: Conceptual View

Yiannis Verginadis¹, Ioannis Patiniotakis¹, Nikos Papageorgiou¹, Roland Stuehmer²

² FZI Forschungszentrum Informatik, Karlsruhe, Germany
stuehmer@fzi.de

¹ Institute of Communications and Computer Systems,
National Technical University of Athens,
{jverg, ipatini, npapag}@mail.ntua.gr

Abstract. In this paper, we present the conceptual architecture of a highly scalable federated platform for processing vast number of events coming from distributed sources, in order to detect interesting situations that lead to service adaptation recommendations. The approach presents an Event Marketplace, a platform for mediating between event providers and (complex) event consumers in very large and heterogeneous environments which is enriched with timely reaction capabilities, through situational-driven service adaptations. We focus on an approach for detecting real-time interesting situations that lead to service adaptations.

Keywords: Event Marketplace, Distributed Systems, Service Adaptation

1 Introduction

Adaptation as one of the basic phenomena of biology is the evolutionary process whereby a population becomes better suited to its habitat [1],[2]. The notion of adaptation has been extensively used, especially nowadays, in the computer science domain. It is considered as one of the most desired functionalities of today's highly dynamic, distributed and ubiquitous environments in the service-oriented setting. The need for highly flexible services that can be orchestrated in order to provide certain behaviors and at the same are able to react and adapt inside their "habitat" (i.e. change based on the context and events that formulate the imprinting of a highly dynamic service environment), is considered to be a desired but difficult to achieve fact.

The recent expansion of the "everywhere" deployment of the wireless sensors networks (part of the Internet of Things) introduces the information from stationary or moving objects in the global service adaptation task. Let us consider the following scenario as an example of ubiquitous interaction between services resulting in a more personalized and adapted service execution:

"Paul is a businessman who has been flying from Paris to New York. He used the entertainment service on board, but hasn't finished watching the movie before the

landing. Two hours later he is entering his room in the downtown hotel he booked earlier and wow: the room entertainment service is ready to play the movie Paul was watching in the plane – of course only the unfinished part.” Such a scenario involves the ubiquitous interaction between services, the processing of dispersed events and the actual adaptation of service execution in a way that satisfies the customer. All these constitute multidimensional problems. Let us name just a few challenges. First of all, such an interaction cannot be modeled in the design time, since it is not possible to predict all interesting interactions in advance. Secondly, a common understanding between different actors in a heterogeneous environment is requested. Next, it is challenging to ensure a proper (on time, complete and relevant) delivery of interesting information in a large scale distributed environment.

In this paper we present the concept and the architecture of a platform that can satisfy these requirements (section 2). The platform uses its cloud-computing nature to be “elastic” and operate in the “pay as you go” mode. We can consider the platform as a kind of the Event marketplace (similar to the notion of the service marketplace) where events coming from different producers can be arbitrary combined by different event consumers. In addition, we focus and give details about a core part of this platform that recommends changes (adaptations) of services’ configurations, composition or workflows, in order to overcome problems and achieve higher performance by reacting to real time events (i.e. formulating an interesting situation) at the right time and in the right way (section 3). In section 4, we present the related work of our approach, whereas section 5 contains concluding remarks.

2 Proposed Conceptual Architecture

The initial conceptual architecture for our platform is depicted in Figure 1. In this section we introduce the components and present their main functionalities.

The Distributed Service Bus (DSB) provides the service oriented architecture (SOA) and event driven architecture (EDA) infrastructure for components and end user services. It acts as the basis for service deployments, and processes (BPEL, BPMN), routing synchronous and asynchronous messages from services consumers to service providers. Based on the principles of the system integration paradigm of Enterprise Service Bus the DSB is distributed by nature.

The Governance component allows users to get information about services and events, as well as specifying QoS requirements as service level agreement (SLA) contracts using the WS-Agreement standard [3]. The Governance component extends a standard Service-based governance tool (OW2 Petals Master [4]) by adding governance mechanisms for event-based systems. Its role is to provide ways to govern services and events. It provides standards-based APIs and a graphical user interface.

The Event Cloud provides storage and forwarding of events. The role of the Event Cloud is a unified API for events, real-time or historic. To that end, it contains a peer-to-peer network to store histories of events durably in a distributed fashion. In the same way the list of subscribers is distributed across the peers to notify subscribers if a given new event is stored at any node and a corresponding matching subscription exists in the system. Subscriptions may use a simple set of operators such as

conjunctive queries which can be evaluated efficiently on a single peer. More complex queries are executed in the DCEP component.

The DCEP component (Distributed Complex Event Processing) has the role of detecting complex events and reasoning over events by means of event patterns defined in logic rules. To detect complex events, DCEP subscribes to the Event Cloud for any simple event defined in the event patterns at a given point in time. DCEP supports traditional event operators such as sequence, concurrent conjunction, disjunction, negation etc., all operators from Allen's interval algebra [5] (e.g., during, meets, starts, finishes etc.), window operators, filtering, enrichment, projection, translation, and multiplication. Out-of-order event processing is supported (e.g. events that are delayed due to different circumstances such as network anomalies).

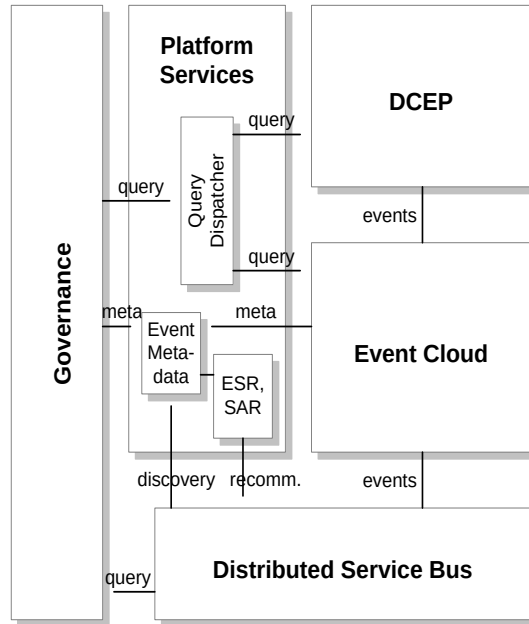


Fig. 1. Conceptual Architecture

The Platform Services component incorporates several functional additions to the platform as a whole. The Query Dispatcher has the role of decomposing and deploying user subscriptions in pieces supported by the Event Cloud and DCEP respectively, taking into account the expressivity supported by the two target components. The Event Metadata component stores information about events, such as source descriptions, event type schemas, etc, to enable the discovery of relevant events for an event consumer and to provide data to the subscription recommender. The ESR and SAR components form the Event Subscription Recommender (ESR) and Service Adaptation Recommender (SAR). ESR will recommend or perform dynamic subscriptions for complex events based on situations detected from semantic-enabled events and the context of subscribed services. Thus, ESR will provide assistance to services that will have the option to be subscribed to specific

events at the “right time” without the services having complete knowledge about the supply in the marketplace at a given time. SAR is thoroughly discussed in the next section of this paper.

3 Service Adaptation Recommender (SAR)

The objective of SAR is to suggest service administrators, changes (adaptations) of their services’ configurations, composition or workflows, in order to overcome problems or achieve higher performance. Based on recognized situations, SAR will be able to define adaptation pointcuts (points in a service flow that need to be adapted as a reaction to a certain situation) and advices (what to adapt and how based on a number of service adaptation strategies). In this section, we present technical requirements that apply for Service Adaptation Recommender (SAR) software component along with its initial conceptual view (figures 2 and 3).

One of the basic capabilities SAR feature is the situation awareness and detection functionality. This corresponds to the ability to sense situations relevant to service objectives and operation and the ability to track transitions between situations, by processing events and contextual service information. We consider complex events, detected in real time, as a way to signify situations that may require adaptation and we plan to enrich them with contextual information for defining dynamically what modifications are needed. Based on situational awareness module, SAR will be able to make intelligent recommendations for service adaptations (figure 2). It will improve service performance by detecting problems that need to be resolved (e.g. underperforming services, or suboptimal service workflows for the given situation) and by providing adaptation advices to be implemented in the appropriate workflow places (e.g. service tasks that need reordering, alteration or substitution etc.).

We have defined some general requirements that apply here. Firstly, it is important that SAR will be able to register for simple events as well as for complex events that carry combined information (e.g. the last 20 minutes, there is a 5% radiation increase). Both simple and complex events need to carry semantics that will allow further processing and reasoning. The federated middleware of our platform must also guarantee that SAR will receive all events that they have been subscribed to. Failure to deliver an event to SAR may lead to the non-detection of a new situation. For reasoning purposes domain knowledge is needed, which may be comprised of an event ontology, a context and a situation model. Scalability is another general requirement that applies to SAR, as the ability to cope with and extract valuable information from a “burst” of events, is imperative for detecting interesting situations. Finally, since our platform will be federated, it is important to detect global situations across the several service busses (i.e. Distribute Service Bus).

Since, SAR will be the dedicated software component to suggest changes (adaptations) of services’ configurations, it needs to leverage domain knowledge for processing contextual information (e.g. preferences) of services and for analyzing service composition information (Fig. 2 - “Service Analyzer”). SAR also needs the ability to comprehend semantics carried by events, reasoning event semantics with service information, in order to detect relevant situations or trigger situational

transitions (Fig. 2 - “Situation Awareness Module”). By acquiring all of the above it will detect and define appropriate service adaptation solutions as timely answers to situations that dictate reactions (Fig. 2 – “Adaptation Recommender”). Regarding DSB, SAR needs subscription/unsubscription capabilities from event sources, in response to situation transitions, as well as query capability to event storage for past events and historical service data in order to identify similar past cases (“Collaborative Filter”).

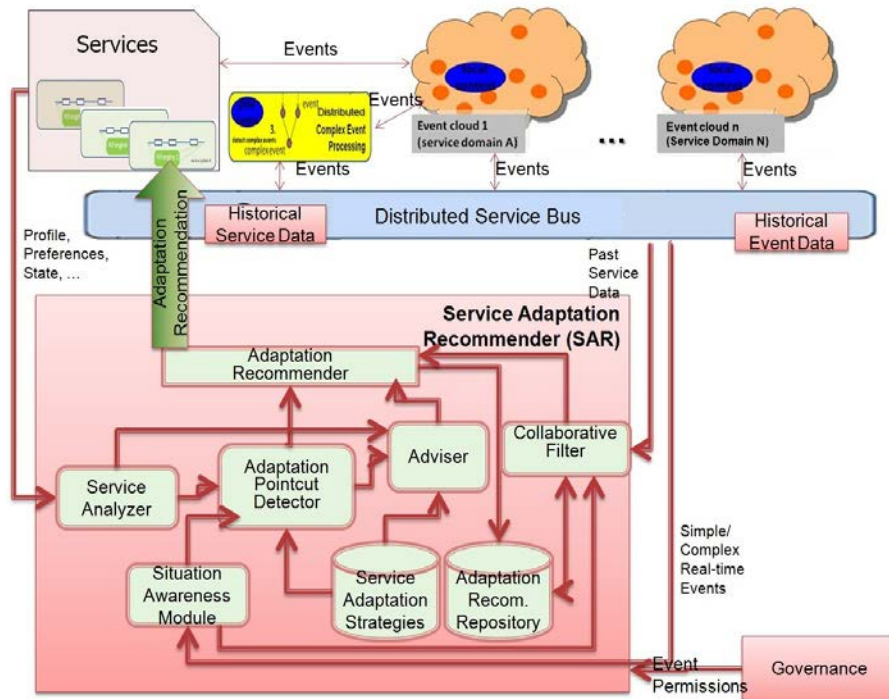


Fig. 2. SAR: Conceptual View

Service Adaptation Recommender (SAR) is considered as a core part of our platform and aims to provide recommender services for reacting at the right time and in the right way. Service Adaptation Recommender (SAR) will suggest changes of services configurations in order to overcome problems or improve service performance by:

- retrieving and analyzing service preferences/context (“Service Analyzer”)
- analyzing service composition information (“Service Analyzer”)
- deducing the situational model for the service using its semantics, composition, and preferences (“Situation Awareness Module”)
- defining where in process flow we might need an adaptation (“Adaptation Pointcut Detector”)

4 Related Work

Recently, there has been a significant paradigm shift towards real-time computing. Previously, queries against databases and data warehouses were concerned with looking at what happened in the past. On the other hand, complex event processing (CEP) is concerned with processing real-time events, i.e., CEP is concerned with what has just happened or what is about to happen in the future. Complex event processing is a very active field of research and is being approached from many angles [6]. Distributed complex event processing approaches circumvent resource limitations by taking advantage of pre-existing (shared) or dedicated network infrastructure. The Padres system [7] is such a distributed approach relying on a pre-existing network of brokers. A broker may be an end-point for publishers (event sources) and subscribers (event sinks) to access the network. More importantly it is the task of the brokers to match events to the available subscriptions. The set of all interconnected brokers forms an overlay network across the underlying infrastructure. A drawback of Padres is the event pattern language used in its subscriptions. It is limited to handle only key-value maps for events and the set of event operators is very limited, for example, temporal relationships between events cannot be expressed declaratively but must be expressed on a timestamp-arithmetical level.

S-Cube [8] as the most prominent Network of Excellence in service adaptation, points out the evolution and adaptation methods and tools as keys to enable service-based applications (SBAs). Following the S-Cube's terminology, the term adaptation refers to the modification of a specific instance of a system during run-time (e.g. re-execution of a unavailable service or a substitution of a unsuitable service). Nowadays, several efforts that try to cope exactly with this issue point to Aspect-Oriented Programming (AOP), as a novel way to weave alternative actions in business processes at run-time. AOP has been proposed as a technique for improving the separation of concerns in software systems and for adding crosscutting functionalities without changing the business logic of the software. One of the most recognizable approaches for service adaptations using AOP is the work in [9] where the AO4BPEL is introduced. AO4BPEL is an XML-based language that creates a wrapper around the BPEL and has the ability to weave aspects at runtime to business processes. Aspects consist of one or several pointcuts and advices. AO4BPEL is based on XPath [10], which is used to select activity join points (i.e., points corresponding to the execution of activities) and internal join points (i.e., points inside the execution of activities such as the point where the outgoing message of an invoke activity is generated). An advice is the new behaviour to be included at a join point and contains the new code to be executed. SAR will use and extend such AOP based approaches for semantically and dynamically weaving changes in service based systems.

5 Conclusions

We presented a novel approach for large scale, context-driven and quality-aware distributed event processing and we focused in the conceptual view of a service

adaptation recommender for coping with the challenges of rapidly changing distributed service based systems.

This research has been performed in the scope of a research project the vision of which is to develop and validate an elastic and reliable architecture for dynamic and complex, event-driven interaction in large highly distributed and heterogeneous service systems. Such architecture will enable ubiquitous exchange of information between heterogeneous services, providing the possibilities to adapt and personalize their execution, resulting in the so-called situational-driven adaptivity. We plan to test and validate our platform and specially its situational driven reaction capabilities, in a nuclear crisis management scenario and in a smart taxi service system. Both these use cases are characterized by dispersed event sources and are considered to create dynamically changing environments that dictate for distributed event processing and service adaptations.

References

1. Martin, E.A.: The Oxford Dictionary of Science, 6th edition. Oxford University Press, ISBN-13: 9780199561469 (2010)
2. Williams, G.C.: Adaptation and natural selection: a critique of some current evolutionary thought. Princeton, NJ: Princeton Univ. Press (1966)
3. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S. and Xu, M.: Web Services Agreement Specification (WS-Agreement). <http://www.gridforum.org/Meetings/GGF11/Documents/draft-ggf-graap-agreement.pdf> (2004)
4. Petals Master SOA Governance Solution, <http://petalsmaster.ow2.org/>
5. Allen, J. F.: Maintaining knowledge about temporal intervals. *Commun. ACM* 26(11), 832—843 (1983)
6. Luckham, D. C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2001)
7. Fidler, E., Jacobsen, H.-A., Li, G., Mankovski, S.: The padres distributed publish/subscribe system. In 8th International Conference on Feature Interactions in Telecommunications and Software Systems, pp. 12—30 (2005)
8. S-Cube, <http://www.s-cube-network.eu/>
9. Charfi, A., Mezini, M.: Ao4bpel: An aspect-oriented extension to bpel. *World Wide Web*, vol. 10, no. 3, pp. 309—344 (2007)
10. XML Path Language (XPath), <http://www.w3.org/TR/xpath/>

Collaboration in open source software development: the ALERT approach

Ljiljana Stojanovic, Jun Ma, Sinan Sen

FZI Karlsruhe
76131 Karlsruhe, Germany
{name.family name}@fzi.de

Abstract. In order to participate effectively in the development activities, FLOSS community members need to understand who is working on what in the community and how their work affects other community members. Current approaches focus on managing information integration, but not active interaction (coordination) between information sources and sinks. In this paper we propose an approach to improve FLOSS coordination by maintaining awareness of community activities through real-time, personalized, context-aware notification. We propose an active collaboration platform, i.e. a virtual actor would interact with other developers, process and recognize various kinds of interactions; suggest actions on the basis of these and remember and bring past interactions into the developers' attention, thus enabling developers to work better together.

Keywords: Event processing, FLOSS, collaboration

1. Introduction

Large scale software development is an inherently collaborative, team based process, and hence requires **coordination**¹ [3] and control in order to make it successful. The advent of global distribution of software development has made managing this collaboration more difficult due to various reasons including physical distance, differences in time, cultural differences etc. Although research on global software development argues the use of **communication** technologies (e.g., e-mails) to alleviate problems caused by separation of workers in time and space, studies have often found them to be not as effective as publicized [7]. The main problem is that, in order to enhance coordination, communication² must be efficient³ (i.e. timely and clear) and effective⁴ (i.e. targeted)

¹ Coordination is the act of integrating each task with each organizational unit, so the unit contributes to the overall objective

² Communication is a mediating factor affecting both coordination and control.

³ Communication is efficient, if information is moved in timely and cost-effective ways and without any barriers except those considered necessary for good coordination.

⁴ Communication is effective if information reaches people who can make productive use of it.

In free/libre open source software (FLOSS) development projects the complexity of managing coordination is **multiplied [1]**. The reasons are manifold.

First, FLOSS communities use a **variety** of communication tools, including mailing lists for technical discussions and support, a bug tracking system for monitoring and fixing bugs, a CVS code repository for storing a common version of the source code. This makes the information pull very heterogeneous, distributed and not well-linked on the basis of its content [2]. Additionally, information and interactions relating to a bug could be located in multiple artefacts with no links. Consequently, a duplicate bug could be filed.

Next, FLOSS communities are very **communication-intensive**. For example, there are about 100 active KDE mailing lists, which accumulate about 1400 messages a day. The **vast amount of information** which is a result of their interactions/communication causes a kind of information overload. There is a lack of coherence that hinders developers in making sense of communications from others, or that produces unintended information filtering or misunderstandings.

Finally, many teams are largely self-organizing, without formally appointed leaders or indications of rank or role, raising the question of how the work of these teams is managed. In fact, deciding what features should be integrated in the release of the software, when and how to empower other code maintainers, or to find most “suitable” developer for fixing a bug, etc. is a challenge and inevitably introduces delays in development. Additionally, FLOSS developers can autonomously decide how and when to contribute to project development. They go through the bugs posted on the bug tracking tool, looking for bugs they can fix. Consequently, a bug could be kept by someone without expertise in the area of the bug, which results in delays-

On the other hand, the quality of the open-source software development process depends very much on when and what is available. For example, if developers do not know in the right moment what others are doing and whether it is necessary to react on, they might be afraid of working on a part of the system for which they are not the only responsible, with the purpose of avoiding conflicts and/or incompatibilities. A lack of common knowledge about the status, authority and competencies of team participants can be an obstacle to the development and results in delays.

Therefore, in order to participate effectively in the development activities, FLOSS community members need to **understand who is working on what in the community and how their work affects other community members**. This knowledge has to be collected in **real time**, as it is created. Such knowledge allows people to coordinate work effectively, anticipate other members’ actions, discuss tasks and locate help [4].

In this paper we propose an approach for improving collaboration between open source developers. The approach is based on context-aware notification, event-driven processing and real-time interactions.

The paper is structured as follows: In section 2 we identify research challenges for improving collaboration among developers during bug resolution process. The requirements for an active collaboration platform have been discussed in section 3. The conceptual architecture of the proposed system is introduced in section 4. Section 5 contains concluding remarks.

2. Research challenges

In this section we discuss research challenges in improving collaboration in bug resolution. We focus on the bug resolution activity, since bug resolution is one of the most difficult software development activities and yet crucial to the quality of the software produced. Developers should feel more free to fix a bug, because they know whether they are the most knowledgeable for resolving a bug and that they will be informed in case of potential conflicts as they happen (i.e. in real time). In the same way, bugs with possible side effects (e.g., duplicates) can be early spotted and tedious conflict prevented.

The following challenges have been identified:

1. **Awareness** – We distinguish between task-oriented and social awareness. Social awareness focuses on humans (their actions, presence, context etc.). Task-related awareness focuses on artifacts (different types of changes in and to them, including creation and deletion; who has made the changes, consequences of the changes etc.) The objective of the awareness mechanism is to help developers to perceive on-going artifact changes and coordinate actions to minimize the development effort.

One of most challenging scenarios for *task awareness* and at the same time, a long-awaited feature for many development teams is **bug duplication detection**. Duplicated issues are a serious time drain, since the amount of effort spent to attend and resolve every issue submitted to the project is by no means negligible. For example, as of February 1, 2006, duplicate reports made up 36% of all reports in the Firefox repository, 17% of Eclipse Platform reports, 14% of Apache 2.0 reports and 13% of Fedora Core reports. These percentages correspond to large numbers of reports: 14159 for Firefox, 9181 for the Eclipse platform, 422 for Apache 2.0 and 4792 for Fedora Core [8]. There exists no automated means (actually, not even *any* kind of tool) facilitating the identification of issues that have been already submitted to the project. There is a need to filter out information from different sources (in particular, starting with the BTS), looking for possible duplicated issues, highlighting them for further inspection by team operators/developers.

Additionally, **similarity detection**⁵ with former bugs should be also supported. The idea is to look for common patterns between new submitted issues and previously solved/closed issues, looking for similarities (same keywords, same severity level etc.). The point here is recovering the approaches followed to address similar past issues that can be also useful to address current ones.

2. **Communication** – Developers participate in the open source community based on their understanding of the **semantics of the community** (who talks to whom, who does what, who knows about what), the **content** (which bugs fix which code, how does the code structure fit together, why was this code implemented this way) and the **interactions** (who said what, who did what).

It is necessary to make the **community semantics more explicit** since it would enable new community members to make use of it without having to

⁵ A similar bug is a bug that is a generalization, specialization or is dependent of another bug

acquire it gradually. The “old” community member could also profit from cross-links between distributed, heterogeneous information. Making people’s semantics accessible to machines allows them to process the community archive, mine it, use and present information to the community in a much more **intelligent** and **contextually-appropriate** manner than is currently taking place. For example, it would be useful to find “**tips of the day**”, showing recommendations about interesting pending bugs (most critical, those with the oldest submission times, those affecting most critical modules, etc.). The information that will be shown as well as the way it will be shown should be personalized and context-aware, which will result in additional expressive power, flexibility and usability.

Additionally, in order for a developer’s conception of on-going work in the FLOSS to be **continuously updated**, relevant information should be updated as a change happens. Notifications are currently often triggered by developers, e.g., via emails or message on discussion forums. As this way of notification is often costly and unreliable, the system should support an **effective tool-based notification**, which promises to reduce the communication effort. For example, subscription functionality would provide a service for developers and maintainers to subscribe themselves to streaming feeders notifying about bugs meeting certain criteria (e.g., matching some keywords, submitted under a certain category/severity level or even detection of bottlenecks, i.e. high number of critical/urgent bugs related to a class).

3. **Task assignment** – The community has collective responsibility for fixing bugs, **no person is explicitly assigned to any bug** [2]. Every now and then, for example, before a software release, people will go through the bugs posted on the bug tracking tool, looking for bugs they can fix. This places the burden of **finding bugs** that can be fixed squarely on the **people who can fix them**. Thus, there can be a long delay before the bug gets to the attention of developers who can fix the bug.

Following a (functionally) similar approach to that found in some online shopping stores (remarkably, Amazon.com), it is necessary to act as a **bug advisor**. It should offer an add-on service that would recommend to a certain member of the development team which issues have the highest probability of being of her/his interest, based on previous bug-tracking history for that developer, maybe complementing that information with feedback from the own member, fields of interest/expertise in her/his profile, etc.

In the rest of this paper we describe the requirements for and the conceptual architecture of ALERT system that realizes the previously mentioned challenges.

3. Requirements

The efficient modeling of coordination has been already recognized as a useful means for improving work in open source development teams. According to the recent literature [5], the coordination in a virtual team critically depends on the possibility to respond in a coordinated fashion to urgent, high stakes, and/or novel task demands. This finding calls for, so called implicit coordination [6], based on the

idea of early warning, that will induce discussions and that will lead to auto-coordination. There are four main steps in this process, inspired by **Observe-Orient-Decide-Act** (OODA)⁶ Loop model of human cognition: people **observe** a SITUATION, then **orient** themselves in the corresponding CONTEXT, **decide** about ACTIONS and **act**⁷.

As the goal of the ALERT system is to improve coordination in the context of FLOSSD by increasing the awareness and responsiveness of the developers and enhancing decision making, this can be done through supporting a four-phased process Observation, Orientation, Decision, and Action, inspired by human cognition and presented in Figure 1.

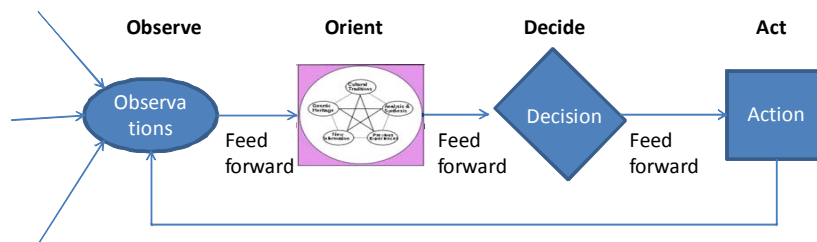


Figure 1: OODA model

Thus, the technical requirements for ALERT can be gathered by analyzing the OODA cycle. In following we discuss very briefly these phases, by emphasizing major challenges in order to get requirements for our technical architecture:

- Observe phase:
 - o **Task:** Scan the environment and gather information relevant for the task at hand.
 - o **Challenge:** Major problem is related to managing awareness of developers: they are usually forced to get information on their own
 - o **General Requirement:** automatic acquisition of relevant information:
 - ALERT system will monitor open source environment and enable proactively alerting (i.e. notification) of developers only about aspects relevant for their work.
- Orient phase:
 - o **Task:** Orientation is how we interpret a situation based on experience, new information, analysis, synthesis, etc. That is, orientation helps to turn information into knowledge.

⁶ http://en.wikipedia.org/wiki/OODA_loop

⁷ "However, the real reason why most people make bad decisions is that they often fail to place the information that we do have into its proper context. This is where "Orientation" comes in. Orientation emphasizes the context in which events occur, so that we may facilitate our decisions and actions. That is, orientation helps to turn information into knowledge. And knowledge, not information, is the real predictor of making good decisions." <http://www.nwlink.com/~donclark/leadership/ooda.html>

- o **Challenge:** Major problem is capturing the right context after observing an interesting situation in order to enable a proper reaction. A common cause is the so-called information overload, i.e. users are faced with lots of irrelevant information around them.
 - o **General Requirement:** Put information that will be extracted from the environment in its proper context, so that for every observation the right interpretation can be found.
 - ALERT will emphasize the context in which information occurs by making community semantics more explicit.
- Decide phase:
 - o **Task:** Consider available options and select a subsequent course of action.
 - o **Challenge:** Usually there are many activities that can be performed in a situation and deciding what to do is a major challenge.
 - o **General Requirement:** Intelligent support for the decision making process that will take into account user and task specificities.
 - ALERT will provide active usage of knowledge by suggesting most-appropriate, task-specific actions for a particular user.
- Act phase – Carry out the conceived decision.
 - o This is out of scope of ALERT as actions have to be done manually.

Once the result of the action is observed, the cycle starts over. Hence, it is a loop. Thus, the loop is actually a set of interacting loops that are kept in continuous operation. The faster and the better the four steps are done, the more enhanced coordination is.

4. Conceptual architecture

The requirements identified in the previous section are summarized in Figure 2. ALERT system has to expose (cf. Events and Information in Figure 2) personalized awareness and coordination information (cf. context-aware notification) in order to derive knowledge (cf. semantic interpretation) and provide active support (cf. intelligent services) for developers who are actively managing work and social dependencies in the open source community.

In the following we are elaborating more on the requirements for each of these phases. These requirements are summarized in Figure 3:

- Observe – Create awareness:

First of all, gathering (and the display) of data should be **disruptive** and **performance of the original system** must not be affected. Secondly, in order to “point out” other’s activity in an effective way, the delay between collection of the data and the relative notification has to be as close as possible to **real-time** (considering network latency). This requires “sensing” interesting information proactively (cf. **Module Sensors** in Figure 3). Additionally, the collection of data has to be **transparent** to the user (no

extra effort must be needed), yet does not have to be invasive (the user does not have to feel “spied”, otherwise he/she will neglect the use of the tool). Information collected should possibly be **fine-grained** both in terms of temporality (where capturing every action made is more precise than grouping them) and granularity of entities (where “*method x has changed*” is more precise than “*file y has changed*”).

From the notification point of view, only users who are interested (**Module Notification Patterns**) in the collected information should be notified (**Module Notifier**). Additionally, users should be informed in a **natural way**: not disturbing, but noticeable and clear. Views (**Module Services – Visualization**) should display activities of other users in an **efficient way** and only **relevant information** should be presented.

- Orient – interpret, understanding communication:

It is necessary to exploit and integrate already existing information. FLOSS community makes use of various kinds of artifacts: Web-based forums (Forums), chat rooms, a bug, issues and patches tracker, a source code repository and change tracker (CVS), source code documentation, etc. To enable the appropriate interpretation of this information, it is necessary to exploit **an explicit representation of the semantic** connections between community artifacts (**Module Ontologies**). This includes **generation of metadata** about various community artifacts as well as **creation of cross-links** between them (**Module Knowledge Extraction**). We cannot expect humans to create and link metadata, so intelligent approaches to gather semantics **semi-automatically** is needed.

Additionally, there is a need to **formalize informal knowledge** which is highly personal, making it difficult to share with others. As it is embedded in individual experience/expertise which is continually changing, it is necessary to construct and dynamically modify developers’ **profiles** (**Module Profiler**).

- Decide – support usage, automatic assignment:

The successful identification and extraction of critical knowledge does not ensure its daily use. It is therefore crucial to provide developers with **means for usage of this knowledge**, i.e. action-oriented services that can take appropriate actions on the basis of the interpretation of knowledge and the current user activity. Firstly, developers need mechanisms to facilitate the **search** for useful knowledge, particularly knowledge that they may not realize exists (**Module Services – Search**). In addition, since developers are often too busy to look for knowledge, proactive services must be provided, actively reminding developers of existence of helpful knowledge. While knowledge search is a user-initiated search, **knowledge recommendation** has to be initiated by the system, without requiring the user to explicitly formulate a query (**Module Services – Recommendation**). Next, different **visual representations** are needed in order to improve transfer of knowledge between the developers (**Module Services – Visualization**). Indeed, various aspects of knowledge need to be visualized in order to enable developers to understand them properly (from different perspectives). Finally, in order to ensure automatic intelligent decision making, a **complex-event processing** is

needed (**Module Services – CEP**). This helps automate intelligent responses to various events and discover parties interested in such complex events.

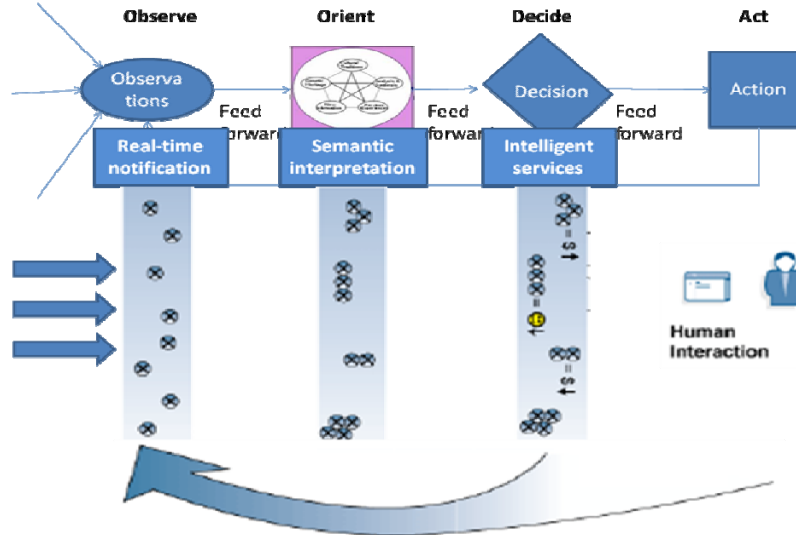


Figure 2: Challenges of the OODA model relevant for ALERT

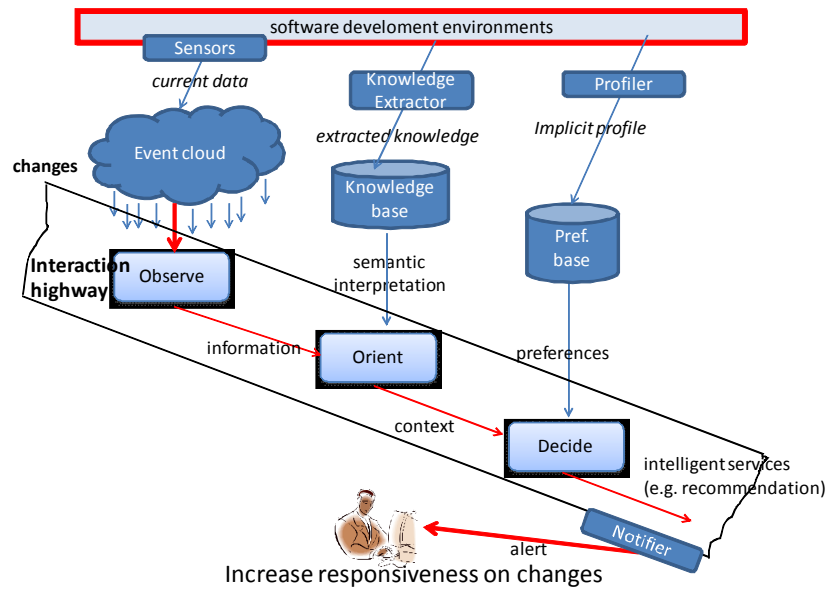


Figure 3: ALERT's response to the OODA requirements

Based on these requirements we have designed the conceptual architecture of the ALERT system as depicted in **Figure 4**.

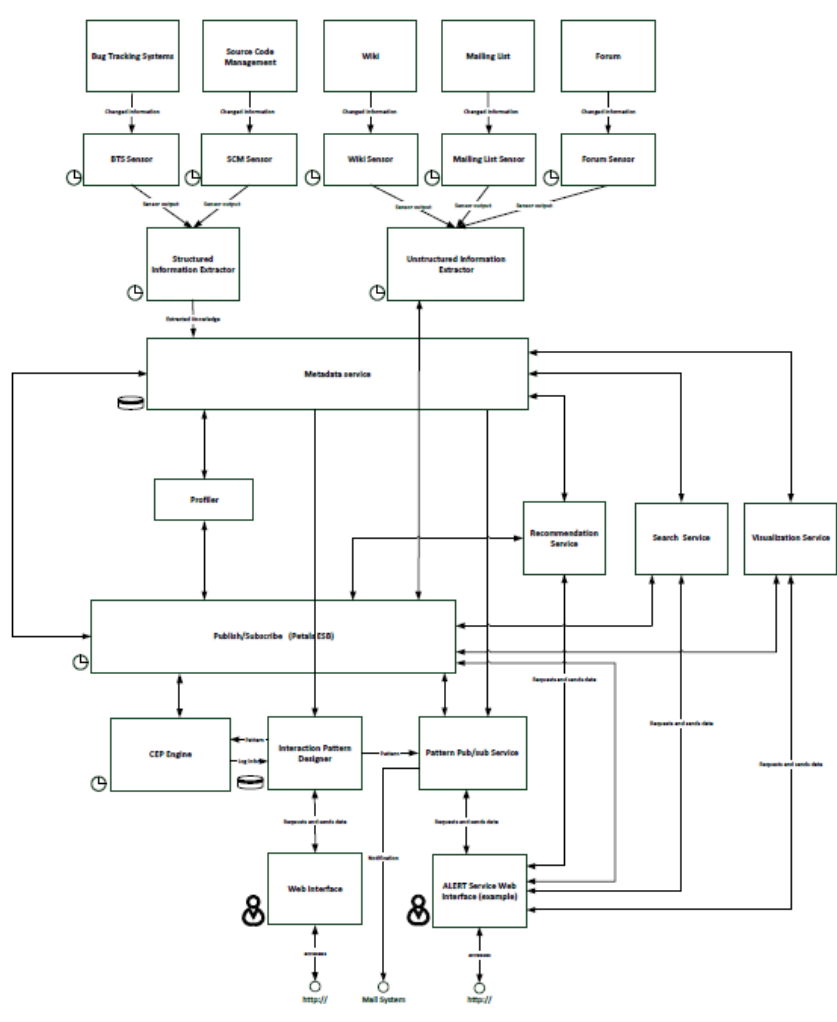


Figure 4: Conceptual architecture of the ALERT system

The conceptual model of ALERT is based on the approach that information about cooperative actions that are relevant for the awareness can be captured through the generation of events. **Events** are generated by sensors that are associated with actors, tools, or any other objects that constitute or influence a cooperative environment.

The reception and generation of activity events is enabled through the realization of lightweight **sensors**. Different sensors should be provided by the infrastructure for the integration into other applications, such as artifact change notifications, or information monitoring agents. Additional sensors should be realizable without a large effort. ALERT will provide sensors for many structured sources (e.g. issue tracking systems) and unstructured sources (e.g. discussion forums).

After creation, an event is transmitted (i.e. **published**) to the interaction highway. The notion of interest profiles is introduced as a method to specify the awareness information someone is interested in. Thus we have adopted that notion and concept for the event distribution and **notification** method of the ALERT environment.

Another important task of this highway is **enrichment** of event information. Event information is interpreted by using background knowledge relevant to bug resolution activity. It requires **an ontology** that describes the structure of the project and provides a basis for determining how artifacts are related as well as its **knowledge base** which does not replace existing sources, but to help augmenting and leveraging the existing information. ALERT provides two ways for creating knowledge base:

- Off-line knowledge acquisition (acquisition of existing knowledge): Potentially useful knowledge already available in existing systems will be made accessible. Information extraction and data mining methods will be used to capture semantics for structured and unstructured sources respectively.
- On-line knowledge acquisition (acquisition of new knowledge): Knowledge base will be **continuously updated** as ALERT will collect artifact change information from all existing sources. Change information will be grouped, classified and filtered to reduce a possible cognitive overload.

ALERT will support distributed team members in providing and receiving awareness information on the task-oriented progress, but also on the social activities. This will be realized by dynamically capturing developers' profile and **clustering of developers based on their profile**.

Event information semantically enriched and interpreted in the context of the knowledge base will be used to trigger not only developers but also some advanced services that will be provided by ALERT. The ALERT collaboration platform will have **two layers of services**: The top layer will consist of **task-specific services** which help developers perform tasks by formulating information needs and resolving these needs through querying, matching, visualizing, reasoning with and exchanging information with other services. The effectiveness of such collaboration platform will increase exponentially as more services become available. The bottom layer will consist of **information-specific services** which provide intelligent access to a heterogeneous collection of information sources. Different "combination" of these services will alleviate the task assignment process: By matching the actual task (i.e. a new issue) and expertise of developers it is possible to create chance encounters. These encounters support core developers in finding "coworkers" with required expertise for a given task and thus can facilitate a task assignment process.

5. Conclusion

In this paper we described the ALERT system that will improve **FLOSS coordination** by **maintaining awareness** of community activities through **real-time, personalized, context-aware notification**. Additionally, instead of being a passive participant in all open source processes, it could be possible to have an **active entity**, a virtual actor, working alongside the developers. The ALERT system will act as an

an **active collaboration platform**, i.e. a virtual actor would interact with other developers, process and recognize various kinds of interactions, suggest actions on the basis of these and remember and bring past interactions into the developers' attention, thus enabling developers to work better together.

Acknowledgement

The authors would like to thank to all the members of the ALERT consortium for their collaboration in the elaboration of this report. Research for this paper was partially financed by EU in the ICT-258098 project "ALERT".

References

- [1] A. Wiggins, Coordination Dynamics in Free/Libre Open Source Software Development, Doctoral Consortium at the Fifth International Conference on Open Source Software (IFIP 2.13). Skövde, Sweden, 3 - 6 June, 2009, <http://www.andreawiggins.com/research/awigginsOSS2009dc-final.pdf>
- [2] A. Ankolekar, Towards a SemanticWeb of Community, Content and Interactions, PhDThesis, Carnegie Mellon University, CMU-HCII-05-103, 2005.
- [3] E. Carmel, R. Agarwal, Tactical Approaches for Alleviating Distance in Global Software Development. In IEEE Software, vol. 18, no. 2, pp. 22-29, March/April 2001.
- [4] C. Gutwin, S. Greenberg, A descriptive framework of workspace awareness for real-time groupware, Journal of Computer-Supported Cooperative Work (JCSCW), 3-4:411–446, 2002.
- [5] L. Beng-Chong, K. Klein, Team mental models and team performance: A field study of the effects of team mental model similarity and accuracy, Journal of Organizational Behaviour 27, pp- 403–418, 2006.
- [6] C. Godart et al., Asynchronous coordination of virtual teams in creative applications (co-design or co-engineering): requirements and design criteria, Proceedings of the workshop on Information technology for virtual enterprises, pages: 135 – 142, 2001.
- [7] Chintan Amrit, Jos van Hillegersberg, Frank Harmsen, 1st International Workshop on Tools for Managing Globally Distributed Software Development (TOMAG 2007) <http://doc.utwente.nl/60288/1/Hillegersberg07Tomag.pdf>
- [8] L. Hiew, Assisted Detection of Duplicate Bug Reports. Master's thesis, University of British Columbia, Canada, 2006. http://www.cs.ubc.ca/grads/resources/thesis/Nov06/Hiew_Lyndon.pdf